ALMA MATER STUDIORUM
UNIVERSITÀ DI BOLOGNA

Laboratorio di
Sicurezza Informatica

Esercitazione: Network Intrusion Detection Systems

Andrea Melis
Marco Prandini
Dipartimento di Informatica – Scienza e Ingegneria

Agenda

- Cenni/Definizioni NIDS
  - Definizione
  - HIDS vs NIDS
  - Tecniche
- Suricata
  - Ping alert
  - Facebook alert
  - Shellshock alert
  - Esercitazione per casa

NIDS

- Come discusso in precedenza, un sistema di rilevamento delle intrusioni è un'applicazione hardware o software che rileva e avvisa gli amministratori quando viene rilevata un'attività dannosa.

- NIDS si concentra principalmente sul monitoraggio e l'analisi del traffico di rete, sia real-time che offline.

Anomaly Based vs. Signature Based IDS

- Anomaly based
  - Monitora il traffico di rete
  - Tiene traccia dei modelli di traffico e delle informazioni per ottenere dati di riferimento
  - Se viene rilevata una deviazione nel comportamento della rete, l'IDS rileverà un attacco.
  - Elevato rischio di falsi positivi
- Signature based
  - Il database delle firme degli attacchi viene mantenuto localment
  - Confronta il traffico con il database
  - Se viene trovata una corrispondenza, viene inviato un avviso
  - Richiede aggiornamenti costanti

HIDS vs NIDS

- NIDS
  - Monitora tutto il traffico sulla rete
  - Utile per monitorare sistemi non critici.
- HIDS
  - IDS personalizzato per un server specifico
  - Essere più vicini all'host consente maggiori possibilità di rilevamento
  - Impedisce l'installazione all'interno della rete di minacce quali Trojan e backdoor

Suricata

- Suricata è un motore di rilevamento delle minacce di rete gratuito e open source, veloce e robusto.

- Il motore Suricata è in grado di eseguire rilevamento delle intrusioni in tempo reale (IDS), prevenzione delle intrusioni (IPS), monitoraggio della sicurezza di rete (NSM) e l'elaborazione dei pcap offline.
- Suricata ispeziona il traffico di rete utilizzando regole potenti ed estese e dispone di un potente supporto di scripting Lua per il rilevamento di minacce complesse.

- Suricata è abilitato a interagire con formati di input e output standard come YAML e JSON,interagisce con strumenti come Splunk, Logstash / Elasticsearch, Kibana e altri database.

Suricata (breve storia)

- Brainstormed da Matt Jonkman, Will Metcalf e Victor Julien
- Nov 2007 - Prime linee di codice scritte da Victor (VIPS)
- 2009 - Il nome viene cambiato in Suricata, il suricata infatti era la mascotte del gruppo.
- Nome del genere latino per Meerkat: Suricata
- Forkato da Snort, suricata è ormai tra i principali tool di network monitoring e Instrusion Detection System

Suricata principali funzionalità

- Multithreading
- Protocol Parsing w/ buffers (http, dns, tls, smb, etc)
- HTTP, DNS, SMB Json Logs
- File Extraction
- IP reputation
- Lua Scripting to perform complicated rule detection logic
- Backed by non-profit (can’t be sold/bought out)
- Netflow Generation
- SMTP Support
- Native IPv6

Suricata principali funzionalità

- Analisi offline dei file PCAP
- Registrazione del traffico utilizzando PCAP logger
- Modalità socket Unix per l'elaborazione automatizzata
- Rilevamento automatico del protocollo
- Eventi JSON e output di avviso
- -
- Logstache, ecc.
- http://suricata-ids.org/features/all-features/

Suricata, guida e installazione

- Documentazione ufficiale, fatta molto bene
  - https://suricata.readthedocs.io
- Installazione dai sorgenti
- Ubuntu con ppa
  - sudo add-apt-repository ppa: oisf / suricata-stable
- Molte procedure dettagliate di installazione per i sistemi comuni
  - https://redmine.openinfosecfoundation.org/projects/suricata/wiki/Suri
- Anche su Windows!

Nella macchina del laboratorio suricata è già installato e configurato per iniziare a lavorarci! :)

Suricata, guida e installazione

- Installazione sulle macchine del laboratorio
  - sudo apt update
  - sudo apt -y install suricata

Suricata, configurazione

- Il file di configurazione è /etc/suricata/suricata.yaml
- La configurazione di default è sufficente per iniziare a testarlo e usarlo
- Potete avere più configurazioni in cascata
- Ogni sezione del file di configurazione principale è commentata efficacemente.
- https://suricata.readthedocs.io is your friend

Suricata, configurazione

```
##
## Step 1: inform Suricata about your network
##

vars:
  # more specifc is better for alert accuracy and performance
  address-groups:
    #HOME_NET: "[192.168.0.0/16,10.0.0.0/8,172.16.0.0/12]"
    HOME_NET: "[192.168.0.0/16,10.0.0.0/8,172.16.0.0/12]"
    #HOME_NET: "[192.168.0.0/16]"
    #HOME_NET: "[10.0.0.0/8]"
    #HOME_NET: "[172.16.0.0/12]"
    #HOME_NET: "any"

    EXTERNAL_NET: "!$HOME_NET"
    #EXTERNAL_NET: "any"

    HTTP_SERVERS: "$HOME_NET"
    SMTP_SERVERS: "$HOME_NET"
    SQL_SERVERS: "$HOME_NET"
    DNS_SERVERS: "$HOME_NET"
    TELNET_SERVERS: "$HOME_NET"
    AIM_SERVERS: "$EXTERNAL_NET"
```

Default files

- File di configurazione: suricata.yaml
- Regole o Signatures, presenti nella sotto-cartella rules
  - decoder-events.rules
  - dns-events.rules
  - files.rules
  - http-events.rules
  - smtp-events.rules
  - stream-events.rules
  - tls-events.rules

Regole suricata

- Cosa è una regola suricata?
- Consideriamo il seguente traffico

```
POST /generate.php HTTP/1.1
User-Agent: DetoxCrypto2
Content-Type: application/x-www-form-urlencoded
Host: detoxcrypto.net16.net
Content-Length: 26
Expect: 100-continue
Connection: Keep-Alive

publickey=sdJoFsAv3jSNMEYxHTTP/1.1 200 OK
Date: Sat, 13 Aug 2016 04:45:30 GMT
Server: Apache
```

Regole suricata

```
alert http $HOME_NET any -> $EXTERNAL_NET any
(msg:"DetoxCrypto Ransomware CnC Activity";
flow:established,to_server; content:"POST"; http_method;
content:"/generate.php"; http_uri; isdataat:!1,relative;
content:”DetoxCrypto”; fast_pattern; http_user_agent;
content:"publickey="; depth:10; http_client_body;
http_header_names; content:!”Referer”; sid:1; rev:1;)
```

Regole suricata

- In riferimento al traffico precedente:

```
POST /generate.php HTTP/1.1
User-Agent: DetoxCrypto2
Content-Type: application/x-www-form-urlencoded
Host: detoxcrypto.net16.net
Content-Length: 26
Expect: 100-continue
Connection: Keep-Alive

publickey=sdJoFsAv3jSNMEYxHTTP/1.1 200 OK
Date: Sat, 13 Aug 2016 04:45:30 GMT
Server: Apache
```

Regole suricata

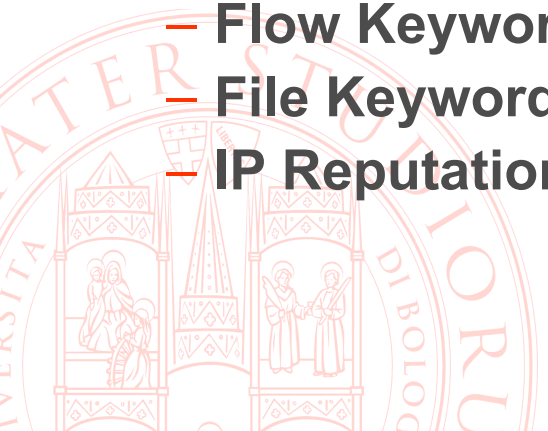
- content:”POST”;
- content:”/generate.php”;
- content:”DetoxCrypto”;
- content:”publickey=“;

```
POST /generate.php HTTP/1.1
User-Agent: DetoxCrypto2
Content-Type: application/x-www-form-urlencoded
Host: detoxcrypto.net16.net
Content-Length: 26
Expect: 100-continue
Connection: Keep-Alive

publickey=sdJoFsAv3jSNMEYxHTTP/1.1 200 OK
Date: Sat, 13 Aug 2016 04:45:30 GMT
Server: Apache
```

Formato delle regole

- Azione: drop, alert, pass, reject, log
- Intestazione: protocol address port direction address port
  - Protocol : ip(all/any), tcp, udp, icmp
  - Address: IPv4, IPv6, $HOME_NET, $EXTERNAL_NET
  - Direction : →(from to) or <> (bidirectional)
- Opzioni
  - Meta-settings #aggiuntive, nessun effetto sull'azione
  - Payload Keywords
  - HTTP Keywords
  - DNS Keywords
  - Flow Keywords
  - File Keywords
  - IP Reputation Keywords



Meta-Settings

- Msg: “hello”
- Sid: (signature id number)
- Rev: (revision of signature)
- Gid: (group type id)
- Classtype: trojan-activity
  - Use classification.config values
- Reference : <type>, <value>
- Priority: 1-255 (normally 1-4, smaller = higher)
- Metadata: “faniofarnogirai”
- content : “abc”

Ricapitoliamo

- Suricata è un IDS con funzionalità di Network Monitoring, active response, e prevention
- Il file di configurazione principale è suricata.yaml e potete specificare le regole da usare, le risposte e le configurazioni per ogni singolo protocollo
- Le regole, che si trovano nella sottocartella /rules sono fondamentalmente di due tipi
  - 1) Predefinite/DI default di suricata, basata sia su comportamenti che su “signature” ovvero specifici attacchi
  - 2) Custom, ovvero definite dall’utente secondo la sintassi precedentemente mostrata
- Le regole hanno il compito di scatenare degli alert o eventi

Utilizzo

- Suricata è già installato sulle macchine del laboratorio. Lanciamo quindi

```
suricata -h
```

- Per ritrovare tutte le opzioni disponibili

Suricata Command Line

- suricata
  - -c <yaml configuration file location>
  - -i <interface to sniff>
  - -s <signatures file> (runs in addition to -c)
  - -r <pcap recording file location>
  - -l <default log directory location>
  - -D }:-)

- Nelle nostre esercitazioni utilizzeremo generalmente suricata in questa modalità

  suricata -c suricata.yaml -i eth0 -vvv

Cambiamo il file di configurazione

- Sostituitamo il file di configurazione assegnato con quello di suricata

  cp FILE_CONF_ESERCITAZIONE /etc/suricata/suricata.yaml

- Analizziamo il file, notiamo in particolar modo:
  - L'insieme delle rules attive
  - L'insieme delle rules commentate
  - I file rules aggiuntivi (c'è specificato un file seclab.rules!)

Aggiungiamo la nostra prima regola

- Come primo test facciamo una regola che “intercetti” qualsisi comunicazione di pacchetti icmp (ping).
- Tutte le regole dell’esercitazione saranno aggiunte al file seclab.rules
- Copiare la prima regola sul file seclab.rules

```
touch /etc/suricata/rules/seclab.rules
cat ping.rules >>  /etc/suricata/rules/seclab.rules
```

- Facciamo partire suricata col seguente comando, l’interfaccia eth1 nel caso in questione è quella dove è associata l’host-only network vboxnet0, modificarla di conseguenza in base alla vostra attuale configurazione

```
suricata -c /etc/suricata/suricata.yaml -i eth1 -vvv
```

Guardiamo gli alert

- A questo punto guardiamo gli alert, in questo modo

```
tail -f /var/log/suricata/fast.log
```

- Dopo di che lanciamo un ping dalla macchina host (o da un altra macchina del laboratorio ad esempio una di quelle clonate). Nel seguente caso viene lanciato il ping dalla macchina Host, verso la macchina VM del laboratorio lungo la rete host-only, l'host ha IP 192.168.56.1 mentre la macchina del laboratorio 192.168.56.3

```
ping 192.168.56.3
```

- Output dell'altert

```
04/28/2021-07:05:55.684973  [**] [1:1000477:1] Ping detected [**]
[Classification: (null)] [Priority: 3] {ICMP} 192.168.56.1:8 ->
192.168.56.3:0
```

Test traffico HTTP

- Immaginiamo ora di voler monitorare l'accesso ad un particolare sito da "dentro" la macchina virtuale del laboratorio
- Questo significa che vogliamo monitorare un preciso traffico Http, con una richiesta ad una particolare pagina
- Immaginiamo di voler monitorare l'accesso al sito facebook.com
- Installiamo la regola data in consegna

```
cat facebook.rules >>  /etc/suricata/rules/seclab.rules
```

- La regola è la seguente

```
alert tls any any -> any any (msg:"SURICATA TRAFFIC-ID: facebook"; tls_sni; content:"facebook.com"; isdataat:!1,relative; flow:to_server,established; flowbits: set,traffic/id/facebook; flowbits:set,traffic/label/social-network; sid:300000001; rev:1;)
```

Test traffico HTTP

- Riavviamo suricata questa volta però sull'interfaccia di NAT della macchina del laboratorio, che nel nostro caso è la eth0

```
suricata -c /etc/suricata/suricata.yaml -i eth0 -vvv
```

- In un altro terminale eseguiamo:

```
wget facebook.com
```

- E osserviamo i log. Domande:
  - Che succede se lanciamo suricata sull'interfaccia eth1 come in precedenza?
  - Verifichiamo che la stessa regola funzioni anche navigano da browser? Cosa notate? Come potremmo analizzare eventuali differenze? (hint. wireshark)

Test Shellshock (dear old friend..)

- Immaginiamo ora di voler inserire delle regole per monitorare un possibile attacco.
- Lavoriamo nuovamente sull'attacco Shellshock che abbiamo visto nella precedente lezione di Wazuh.
- Come per i precedenti test inseriamo la regole

```
cat shellshock.rules >>  /etc/suricata/rules/seclab.rules
```

- A questo punto assicuriamoci che il web server sulla macchina virtuale del laboratorio sia attivo

```
systemctl status nginx
```

```
active...
```

Test Shellshock (dear old friend..)

- A questo punto vogliamo simulare un attacco verso la macchina del laboratorio. Come per il caso del ping facciamo una richiesta dalla macchina host via rete host-only vboxnet0
- Avviamo suricata

```
suricata -c /etc/suricata/suricata.yaml -i eth1 -vvv
```

- Simuliamo un attacco shellshock (nel nostro caso verso 192.168.56.3)

```
curl --insecure IP_MACCHINA_LABORATORIO -H "User-Agent: () { :; }; /bin/cat /etc/passwd"
```

guardare i fast.log di suricata.

Log JSON

- Oltre ai fast log suricata offre anche dei log più “sofisticati” e espressivi in formato JSON.
- Questi log sono nel file /var/log/suricata/eve.json
- Essendo in formato JSON è molto comodo utilizzare dei tool appropriati che possano “parsarlo” in modo adeguato.
- Un esempio è jq:

```
sudo apt install jq
```

- Esempio

```
tail -f /var/log/suricata/eve.json | jq 'select(.event_type="alert")'
```

Suricata offline

- Tutte i test fatti finora sono stati fatti secondo la modalità “runtime”. In questa modalità infatti analizziamo a tempo reale del traffico in entrata o uscita dal nodo di controllo
- Suricata può però lavorare anche in modalità “offline”
- La modalità offline ci permette di analizzare delle tracce di traffico precedentemente catturate. Questa analisi può quindi essere fatta ex-post ad un attacco, oppure semplicemente può essere fatta in sicurezza su una macchina scollegata dalla rete.
- Questa modalità è quindi molto utile per analisi di attacchi una volta che sono stati “effettuati”

Suricata offline

- Il funzionamento della modalità offline è abbastanza semplice.
- Il file di configurazione, le regole e tutte le altre impostazioni NON cambiano
- L'unica cosa che cambia è la modalità di esecuzione di suricata.
- Invece che lanciarlo in "listening" su un'interfaccia di rete suricata viene lanciato dandogli in input il tracciato di un traffico.
- Ma in quale formato? In formato pcap è sufficente!

```
suricata -c /etc/suricata/suricata.yaml -r FILE_PCAP_TRACCIATO
```

Suricata offline, generiamo il pcap

- Facciamo un semplice test.
- Apriamo wireshark (sempre da utente root) e mettiamoci in ascolto sull'interfaccia di rete host-only (nel nostro caso eth1). Questo significa lanciare il "capture" sull'interfaccia eth1
- A questo punto, come per la prima esercitazione lanciamo un ping dalla macchina host (o una delle macchine clonate) verso la macchina del laboratorio.

ping 192.168.56.3

Suricata offline, generiamo il pcap

- A questo punto dovremmo iniziare a vedere del traffico ICMP su wireshark.
- Facciamo andare un po di ping dopo di che eseguiamo stop della cattura.
- A questo punto salviamo il tracciato (File → Salva con nome..) nella home dell'utente sec con il nome tracciato_ping.pcapng

- Abbiamo quindi generato il nostro tracciato, possiamo quindi eseguire il comando enunciato in precedenza.

```
suricata -c /etc/suricata/suricata.yaml -r tracciato_ping.pcapng
```

Guardare i fast log….

Ricapitoliamo

- Abbiamo introdotto suricata, un NIDS molto potente.
- Abbiamo visto la struttura del file di configurazione di suricata
- Abbiamo visto come si crea una regola per suricata, e come sia possibile avviarla.

- Abbiamo fatto alcuni test
  - Ping monitoring
  - Website monitoring
  - CVE monitoring
  - Suricata offline

Esercitazioni per casa

- Prima esercitazione (easy)

  Lo studente deve costruire (e validare) una regola suricata che identifichi il traffico in uscita per il portale netflix.com

  Nota bene:

  La regola va generata con un certo criterio. Che significa? Significa che una richiesta a netflix.com comporta più richieste a più domini!

  Analizzare quindi con wireshark, o altro tool tutti domini coinvolti… Ricordate la lezione su Web Sec!

Esercitazioni per casa

- Seconda esercitazione (medium) (1/2)

  Allo studente è assegnato un file .pcapng

  Il file è il tracciato di un traffico mqtt tra un subscriber e un publisher sul broker mosquitto.

  Compito dello studente è quello di:

  Creare una regola suricata che scateni un alert ogni volta che nel contenuto del pacchetto MQTT ci sia il contenuto “flag”

Esercitazioni per casa

- Seconda esercitazione (medium) (2/2)

  Se eseguita correttamente nei log di suricata dovreste essere in grado di vedere il contenuto dei pacchetti

  Nel contenuto dei pacchetti è possibile trovare “pezzi” di una flag nel formato SEC{qualcosa}, che potete ricostruire e sottometterla (insieme alla regola!) sul portale virtuale.

  ATTENZIONE: Per poter vedere il contenuto dei pacchetti nei log di suricata è necessario abilitare la funzionalità “payload-printable”. Cercare nel file di configurazione di suricata la suddetta feature e abilitarla se disabilitata.